

Exploiting Parameter Domain Knowledge for Learning in Bayesian Networks

Radu Stefan Niculescu

June 2005

THESIS SUMMARY

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

*Submitted in partial fulfillment of the requirements for
the degree of Doctor of Philosophy*

Thesis Committee:

Professor Tom Mitchell, Chair

Professor John Lafferty

Professor Andrew Moore

Dr. Bharat Rao, Siemens Medical Solutions

This research was sponsored by the SRI International under subcontract 03-0002111 for the Department of the Interior, the National Science Foundation under grant nos. CCR-0085982 and CCR-0122581, a Merck Corporation Graduate Fellowship and a generous gift from Siemens Medical Solutions. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: Constraint Optimization, Domain Knowledge, Graphical Models

Abstract

The task of learning models for many real-world problems requires researchers to incorporate problem Domain Knowledge into the learning algorithms because there is rarely enough training data to enable accurate learning of the structures and underlying relationships in the problem. Domain Knowledge comes in many forms. Domain Knowledge about relevance of variables (Feature Selection) can help us ignore certain variables when building our model. Domain Knowledge specifying conditional independencies among variables can guide our search over possible model structures. This thesis presents a theoretical framework for incorporating a different kind of knowledge into learning algorithms for Bayesian Networks: Domain Knowledge about relationships among parameters.

We develop a unified framework for incorporating general Parameter Domain Knowledge constraints in learning procedures for Bayesian Networks by formulating this as a constrained optimization problem. We solve this problem using iterative algorithms based on Newton-Raphson method for approximating the solutions of a system of equations. We approach learning from both a frequentist and a Bayesian point of view, from both complete and incomplete data.

We also derive closed form solutions for our estimators for several types of Parameter Domain Knowledge: parameter sharing, as well as sharing properties of groups of parameters (sum sharing and ratio sharing). While models like Module Networks, Dynamic Bayes Nets and Context Specific Independence models share parameters at either conditional probability table or conditional distribution (within one table) level, our framework is more flexible, allowing sharing at parameter level, across conditional distributions of different lengths and across different conditional probability tables. Other results include several formal guarantees about our estimators and methods for automatically learning domain knowledge.

To validate our theory, we carry out experiments showing the benefits of taking advantage of domain knowledge for modelling the fMRI signal during a cognitive task. Additional experiments on synthetic data are also performed.

Thesis Summary

1 Motivation

Probabilistic Models have become increasingly popular in the last decades because of the need to characterize the non-deterministic nature of relationships among variables describing many real world domains. Among these models, *Bayesian Networks* have received a tremendous amount of interest because of their ability to compactly encode uncertainty about random variables and to efficiently deal with missing data. Another major advantage of Bayesian Networks is that they are relatively easy to interpret by a non-expert, unlike Neural Networks or Support Vector Machines. Applications of Bayesian Networks include medical diagnosis, stock market prediction, fraud detection, intelligent troubleshooting and language modelling.

A *Bayesian Network* is a model that compactly represents the probability distribution over a set of random variables. It consists of two components: a structure and a set of parameters. The structure is a Directed Acyclic Graph where one can think of the edges as cause-effect relationships. The parameters describe how each variable relates probabilistically to its parents. Intuitively, the parameters describe how probable each effect is given a combination of direct causes.

Figure 1 shows a simplified version of a Bayesian Network that can be used for disease diagnosis. Typically, a diagnosis is reached by looking at a combination of risk factors and symptoms. Risk factors like *Smoking* (whether or not the patient smokes), *FHxMI* (whether or not the patient has a family history positive for heart attack), *Pollution* (whether or not the area where the patient lives has high air pollution) can all determine the presence of a disease. Given a disease is present, the patient may or may not show certain symptoms: *Fever*, *Chest Pain*, *Vomiting*.

The task of learning models for many real-world problems requires researchers to incorporate problem *Domain Knowledge* into the learning algorithm because there is rarely enough training data to enable the learning of the structures and underlying relationships in the problem. Domain Knowledge comes in many forms. A domain expert can provide Domain Knowledge about relevance of certain variables, also called Feature Selection, that can help us ignore certain variables when building our model. Domain Knowledge specifying conditional independencies among variables can both guide our search over possible Bayesian Network structures and speed up inference. Both these forms of Domain Knowledge have been extensively studied.

This thesis presents a theoretical framework for incorporating a different kind of knowledge into *Parameter Learning* algorithms for Bayesian Networks: Domain Knowledge about relationships among parameters. *Parameter Learning* in a Bayesian Network is defined as the problem of estimating the parameters of that network from a dataset of training cases. These cases can be either fully or partially observable.

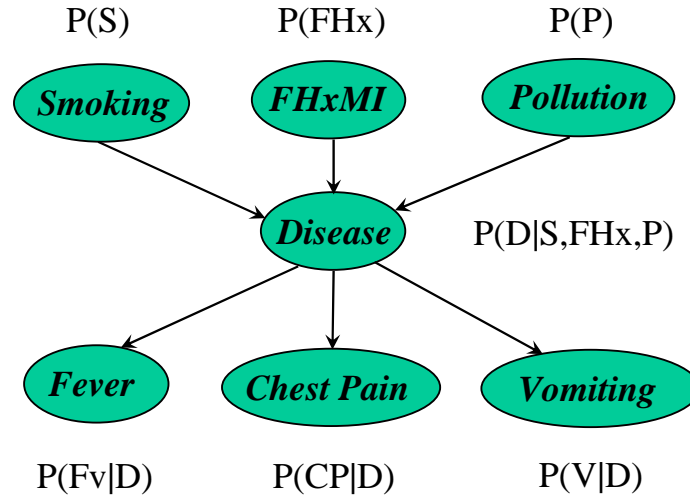


Figure 1: A simplified version of a Bayesian Network which models the interaction between risk factors, diseases and symptoms for the purpose of disease diagnosis in Emergency Room patients. The variables involved are: *Smoking*(S), Family History of Heart Attack *FHxMI* (FHx), *Pollution*(P), *Disease*(D), *Fever*(Fv), *Chest Pain*(CP) and *Vomiting* (V).

To see why one would need to take advantage of *Parameter Domain Knowledge*, consider the network in the above example. In a real world situation, we can have tens of potential risk factors and hundreds of potential symptoms. Also, the *Disease* variable can take values in very large set. With only 20 binary risk factors, the number of parameters of the diagnosis Bayesian Network easily runs in the millions. Unfortunately, clean and complete medical data is extremely hard to come by in a quantity sufficient to allow us to estimate these parameters accurately. However, medical Domain Knowledge is plentiful and it can come directly from physicians or can be extracted from written/online medical material. For example, a doctor may say: *All the other risk factors can be ignored (have little additional influence) when deciding on a diagnosis of Heart Attack given that the patient is a Smoker with a positive Family History of Heart Attack*. Knowledge coming from medical books may state: *A patient with Heart Attack exhibits chest pain and, very frequently, vomiting*. While in the first case, domain knowledge asserts equality of a large number of parameters, in the second case it asserts a deterministic relation between *Heart Attack* and *Chest Pain*.

First, *Parameter Domain Knowledge* can help by shrinking the space in which the parameters can take values. In the case of equality constraints, we achieve dimensionality reduction in this space (as we noticed in the above examples). Inequality constraints can significantly reduce the volume of the feasible region in the space of parameters in the case when this region is bounded. This is the case with Bayesian Networks that model only discrete variables, because each parameter is a probability between 0 and 1. Second, since *Parameter Domain Knowledge* has the effect of

shrinking the space of allowed parameters and since the amount of training data does not change, intuitively one would expect that algorithms that know how to take advantage of Parameter Domain Knowledge will produce lower variance estimators, which would be a great plus when training data is scarce.

Currently, most popular ways of representing *Parameter Domain Knowledge* fall into two categories: Dirichlet Priors and their variants (including Dirichlet Tree Priors, Dependent Dirichlet Priors and smoothing techniques) and Parameter Sharing (HMMs, Dynamic Bayesian Networks, Module Networks, Bayesian Multinetworks, Context Specific Independence, Bilinear Models, Kalman Filters, Object Oriented Bayesian Networks and Probabilistic Relational Models). One of the main problems with Dirichlet Priors and related models is that it is impossible to represent even simple equality constraints between parameters without using priors on the parameters of the Dirichlet Prior, in which case the marginal likelihood can not be computed in closed form anymore and expensive approximate methods are required to perform parameter estimation. A second problem is that it is often beyond the expert's ability to specify a full Dirichlet Prior on the parameters of a Bayesian Network. Parameter Sharing methods can only represent equalities among parameters, but no other, more complicated, constraints. Current models use parameter sharing at either the level of conditional probability table (Module Networks, HMMs) or at the level of conditional probability distribution (Context Specific Independence) within the same table. No such model allows sharing at parameter level of granularity.

The main contribution of this thesis is an unified framework that allows us to incorporate any kind of domain knowledge constraints (that obey certain differentiability assumptions) in parameter learning procedures for Bayesian Networks. We present closed form solutions for several types of Parameter Domain Knowledge which the methods described in this chapter can not represent. We show how widely used models including Hidden Markov Models, Dynamic Bayesian Networks, Module Networks and Context Specific Independence are just particular cases of one of our Parameter Domain Knowledge types, namely the General Parameter Sharing Framework. This framework is able to represent parameter sharing assumptions at parameter level of granularity, which previous models were not able to do. While the domain knowledge presented in this chapter can only accommodate simple equality constraints between parameters, we also derived closed form solutions for Parameter Domain Knowledge types that involve relationships between groups of parameters (sum sharing, ratio sharing). Moreover, we show how to compute closed form Maximum Likelihood estimators when the domain knowledge comes in the form of several types of inequality constraints. Along with our estimators come a series of formal guarantees that show the benefits of taking advantage of the available domain knowledge and also study the performance in the case when the domain knowledge might not be entirely accurate. Finally, we developed methods to automatically learn the domain knowledge, which we illustrate in section 3.

The Thesis Summary will proceed as follows. In section 2 we briefly present our research approach. Section 3 is intended to give the reader an idea of what kind of Parameter Domain Knowledge assumptions we can deal with. That section contains an application of our algorithms on a real world problem of modelling the fMRI signal during a cognitive task. Section 4 summarizes all the contributions of this research.

1.1 Thesis Statement

Standard methods for performing parameter estimation in Bayesian Networks can be naturally extended to take advantage of domain knowledge that can be provided by a domain expert. These new methods can help lower the variance in parameter estimates by reducing the number of degrees of freedom in the space of allowed parameters. While with an infinite amount of training data one would expect standard parameter estimation methods to perform very well, the impact of incorporating Domain Knowledge constraints is quite noticeable when training data is scarce.

2 Research Approach

The derivation of our results relies heavily on optimization and approximation techniques. We will formulate maximum likelihood parameter learning from complete data as a constrained maximization problem. We will solve this optimization problem using Karush-Kuhn-Tucker theorem. This is a generalization of Lagrange Multipliers theorem, which looks for a set of inequality constraints that become equalities at the local optimum. As expected, the system of equations which results from Karush-Kuhn-Tucker theorem may be difficult to solve in closed form in the general case. However, this system has the same number of equations as variables and its solutions can be found using the Newton-Raphson iterative method. For this method to work, we require that our Parameter Domain Knowledge constraints be represented as twice differentiable functions with continuous second derivatives.

The dimensionality of the optimization problem can make the above approach prohibitive. Fortunately, in practice, most given constraints involve only a small fraction of the total number of parameters. In addition, the objective function (likelihood function in general) is nicely decomposable and therefore we are able to split the initial maximization problem into a set of many independent maximization subproblems, each with its own set of constraints. These subproblems have much lower dimensionality and therefore can be solved much easier with the above mentioned method.

There are several approaches to parameter learning: from either a frequentist or a Bayesian point of view, from either complete or incomplete data. The above method performs learning from complete data from a frequentist point of view. In the case of incomplete data, we present several ways to perform Maximum Likelihood estimation based on methods similar with the ones for

complete data. In particular, we notice that extending the Expectation Maximization algorithm for discrete Bayesian Networks in the presence of Parameter Domain Knowledge constraints is just a matter of applying in the M-Step the Maximum Likelihood estimators on the expected counts computed in the E-Step. From a Bayesian point of view, we define *Constrained Parameter Priors* that obey the Parameter Domain Knowledge and show how the normalization constant can be computed via a sampling algorithm. Based on these priors, we then discuss how one can perform Maximum A posteriori estimation and Bayesian model averaging for both complete and incomplete data.

While the above methods work for general constraints, it would be preferable to be able to compute in one step closed form solutions for both the parameter estimators and the normalization constants of the *Constrained Parameter Priors*. Unfortunately, this task is not always possible, simply because of the fact that there is no known closed form solution for polynomial equations of degree higher than four. Three chapters of this thesis are dedicated to the derivation of closed form estimators for several types of domain knowledge. We study Parameter Domain Knowledge constraints for both discrete and continuous variables, with a special emphasis on parameter sharing. However, we also investigate constraints between sets of parameters (sum sharing, ratio sharing). In one of these three chapters, we compute closed form Maximum Likelihood estimators in the case when the domain knowledge comes as inequality constraints. These derivations are performed by directly solving the system of equations that characterize the maximum point instead of resorting to the iterative method.

To validate our approach, we perform experiments on both synthetic and real world data. We compare our models with standard baseline models using the *KL divergence* in the case of synthetic data and the *Average Log Score* (which converges to the negative of cross-entropy on the long run) in the case of real world data.

3 Experiments

Functional Magnetic Resonance Imaging (fMRI) is a technique for obtaining three-dimensional images of activity in the brain throughout time. More precisely, fMRI measures the ratio of oxygenated hemoglobin to deoxygenated hemoglobin in the blood with respect to a control baseline, at many individual locations within the brain. This is often referred to as the blood oxygen level dependent (BOLD) response. The BOLD response is taken as an indicator of neural activity.

An fMRI scanner records a 3D image of the brain as a collection of parallel slices. Each such slice contains a collection of small cells, called voxels. A voxel has a resolution of few tens of cubic milliliters and can contain hundreds of thousands of neurons. In our dataset, there are eight parallel slices for each fMRI snapshot and the dimension of each voxel is few tens of cubic millimeters. Typically, there are ten to fifteen thousand voxels in a human brain. However, only a part of them

are available in our dataset.

During an fMRI experiment, a subject is asked to perform several trials of a cognitive task while the fMRI scanner is monitoring the BOLD signal. It is common for a trial to last few tens of seconds, a snapshot of the brain being captured once or twice per second. A common use for the data collected in these trials is to come up with regions of the brain that are active during the performed cognitive task. A slightly different approach uses the fMRI signal to classify different cognitive states in which the subject may be at different points in time. As it is the case with the Naive Bayes classifier, it is well known that models that perform very well on a classification task may represent poorly the underlying structure of the data.

As opposed to the approaches above, in this section we present a generative model of the activity in the brain during a cognitive task based on parameter sharing assumptions for the Hidden Process Models that describe the fMRI signal. These parameter sharing assumptions are not readily available, but we successfully employ a cross-validation approach to automatically discover clusters of voxels that can be learnt together using Shared Hidden Process Models. We show that our methods far outperform the baseline Hidden Process Model that is learnt on a per voxel basis. Let us start by explaining Hidden Process Models.

3.1 Parameter Sharing in Hidden Process Models

A *Hidden Process Model (HPM)* is a probabilistic framework that predicts the value of a *target variable* X at a given point in time as the sum of the values of certain *Hidden Processes* that are active. This model is inspired from observations of the fMRI signal in the brain when a subject performs a cognitive task. One can think of the target variable as the value of the fMRI signal in one small cube inside the brain (also called a voxel). A hidden process may be thought of as the fMRI activity that happens as a response to an external *stimulus*. For example, a "Picture" process may describe the fMRI signal that happens in the brain starting when the subject is presented with a picture. A "Sentence" process may provide the same characterization for the situation when a subject is reading a sentence. In the real world several stimuli may be active at some point in time and it is conjectured that the observed fMRI signal is the sum of the corresponding processes, translated according to their starting times.

Formally, a *Hidden Process Model* is a collection of time series (also called hidden processes): P_1, \dots, P_K . For each process P_k with $1 \leq k \leq K$, denote by P_{kt} the value of its corresponding time series at time t after the process started. Also, let X_t be the value of the target variable X at time t . If process P_k starts at time t_k , then a Hidden Process Model is predicting the X_t using the following distribution:

$$X_t \sim N\left(\sum_k P_{k(t-t_k+1)}, \sigma^2\right)$$

where σ^2 is considered to be the variance in the measurement and it is kept constant across time. For the above formula to make sense, we consider $P_{kt} = 0$ if $t < 0$. While in the real world it may happen that the subject is presented with the same kind of stimulus multiple times across time, here we make the assumption that each process is active at most once within each example.

In an fMRI experiment, the subject may perform the same cognitive task multiple times and that leaves us with multiple observations about X_t , the value of X at time t . In our framework we denote by X_{nt} the value of X_t and by t_{nk} the starting point of process P_k in example n in the dataset of observations about X , where each observation tracks the value of X across time given a combination of processes that can start at any times. Let N be the total number of observations. Now we can write:

$$X_{nt} \sim N\left(\sum_k P_{k(t-t_{nk}+1)}, \sigma^2\right)$$

While not entirely necessary for our method to work, several assumptions will allow us to make a more compact presentation of Hidden Process Models, based on some characteristics of the fMRI dataset that we are going to use in our experiments. First, we assume that X is tracked across the same time length in each observation. Let T be the length of every such observation (trial). Since we are not modelling what happens when $t > T$, we can also consider that each process has length T . Second, in our fMRI dataset, we know in advance when the stimuli are presented and therefore in our model we assume that t_{nk} , the starting times of the processes, are fully observable.

The same stimuli may influence the activity in multiple voxels of the brain during one cognitive task. For example, looking at a picture may activate many voxels in the visual cortex. The activation in these voxels may be different at each given point in time. Intuitively, that means the same stimulus may produce different hidden processes in different voxels. However, certain groups of voxels that are close together often have similar shape time series, but with different amplitude. In this case, we believe it is reasonable to assume that the underlying hidden processes corresponding to these voxels are proportional to each other. Experiments performed in next subsection will prove that this assumption helps learn better models than the ones that choose to ignore it.

In the above paragraph we explained intuitively that sometimes it makes sense to share the same base processes across several time-varying random variables, but allow for different scaling factors. Formally, we say that time-varying random variables X^1, \dots, X^V share their corresponding *Hidden Process Models* if there exist base processes P_1, \dots, P_K and constants c_k^v for $1 \leq v \leq V$ such that:

$$X_{nt}^v \sim N\left(\sum_k c_k^v \cdot P_{k(t-t_{n_k}^v+1)}, \sigma^2\right)$$

and the values of different variables X^v are independent given the parameters of the model. Here σ^2 represents the variance in measurement which is also shared across these variables.

Next we will study how to efficiently perform Maximum Likelihood estimation of the parameters of the variables X^1, \dots, X^V , assuming that they share their corresponding Hidden Process Model parameters as described above. The parameters to estimate are the base process parameters P_{kt} where $1 \leq k \leq K$ and $1 \leq t \leq T$, the scaling constants c_k^v (one for each variable V and process k) where $1 \leq v \leq V$ and the common measurement variance σ^2 . Let $P = \{P_{kt} \mid 1 \leq k \leq K, 1 \leq t \leq T\}$ be the set of all parameters involved in the base processes and let $C = \{c_k^v \mid 1 \leq k \leq K, 1 \leq v \leq V\}$ be the set of scaling constants. We remind the reader that N represents the number of observations. The log-likelihood of the model is given by:

$$l(P, C, \sigma) = -\frac{NTV}{2} \cdot \log(2\pi) - NTV \cdot \log(\sigma) - \frac{1}{2 \cdot \sigma^2} \cdot \sum_{n,t,v} (x_{nt}^v - \sum_k c_k^v \cdot P_{k(t-t_{n_k}^v+1)})^2$$

It is easy to see that the value of (P, C) that maximizes l is the same for all values of σ . Therefore, in order to maximize l , we can first minimize $l'(P, C) = \sum_{n,t,v} (x_{nt}^v - \sum_k c_k^v \cdot P_{k(t-t_{n_k}^v+1)})^2$ with respect to (P, C) and then maximize l with respect to σ based on the minimum point for l' . One may notice that l' is a sum of squares, where the quantity inside each square can be seen as a linear function in both P and C . Therefore one can imagine an iterative procedure that first minimizes with respect to P , then with respect to C using the Least Squares method. Once we find $M = \min l'(P, C) = l'(\hat{P}, \hat{C})$, the value of σ that maximizes l is given by $\hat{\sigma}^2 = \frac{M}{NVT}$. This can be derived in a straightforward fashion by enforcing $\frac{\partial l}{\partial \sigma}(\hat{P}, \hat{C}, \hat{\sigma}) = 0$. With these considerations, we are now ready to present an algorithm to compute Maximum Likelihood estimators $(\hat{P}, \hat{C}, \hat{\sigma})$ of the parameters in the shared Hidden Process Model:

Algorithm (Maximum Likelihood Estimators in a Shared Hidden Process Model) Let \bar{X} be the column vector of values x_{nt}^v . Start with a random guess (\hat{P}, \hat{C}) and then repeat Steps 1 and 2 until they converge to the minimum of the function $l'(P, C)$.

STEP 1. Write $l(\hat{P}, \hat{C}) = \|A \cdot \hat{P} - \bar{X}\|^2$ where A is a NTV by KT matrix that depends on current estimator \hat{C} of the scaling constants. Minimize with respect to \hat{P} using ordinary Least Squares to get a new estimator $\hat{P} = (A^T \cdot A)^{-1} \cdot A^T \cdot \bar{X}$.

STEP 2. Write $l(\hat{P}, \hat{C}) = \|B \cdot \hat{C} - \bar{X}\|^2$ where B is a NTV by KV matrix that depends on current estimator \hat{P} of the base processes. Minimize with respect to \hat{C} using ordinary Least Squares to get

a new estimator $\hat{C} = (B^T \cdot B)^{-1} \cdot B^T \cdot \bar{X}$.

STEP 3. Once convergence is reached by repeating the above two steps, let $\hat{\sigma}^2 = \frac{l'(\hat{P}, \hat{C})}{NVT}$.

It might seem that this is a very expensive algorithm because it is an iterative method. However, we applied it in our experiments of modelling the fMRI signal during a cognitive task and it turns out it usually converges in 3-5 repetitions of Steps 1 and 2. We believe that the main reason why this happens is because at each partial step during the iteration we compute a closed form global minimizer on either \hat{P} or \hat{C} instead of using a potentially expensive gradient descent algorithm. Next we will experimentally prove the benefits of this algorithm over methods that do not take advantage of parameter sharing assumptions, i.e. the shared Hidden Process Models corresponding to neighboring voxels in the brain when the subject is performing a cognitive task.

3.2 Experimental Setup

We experimented using the *StarPlus* dataset. The StarPlus experiment was designed to engage several different cortical areas, in order to look at their interaction. In this dataset, each subject first sees a sentence(semantic stimulus) for 4 seconds, such as “The plus sign is above on the star sign.”, then a blank screen for 4 seconds, and finally a picture(symbol stimulus) such as

$$\frac{+}{*}$$

for another 4 seconds. At any time after the picture was presented, the subject may press a button for “yes” or “no”, depending on whether the sentence matches the picture seen or not. The subject is instructed to rehearse the sentence in his/her brain until the picture is presented rather than try to visualize the sentence immediately. The second variant switches the presentations of sentences and pictures, and the instruction is to keep the picture in mind until the presentation of the sentence.

In this dataset, the voxels are grouped in 24 ROIs (Regions of Interest, defined based on brain anatomy), each voxel having a resolution of 3 by 3 by 5 millimeters. A snapshot of the brain is taken every half second. In this dataset there are three main conditions: *fixation* (the subject is looking at a point on the screen), *sentence followed by picture* and *picture followed by sentence*. We have 10 trials in *fixation* and 20 in each of the other two conditions. For each trial, we kept 32 (16 seconds) snapshots of the brain.

Our goal is to come up with a model that best explains the activity in the brain when a subject is either reading a sentence or looking at a picture. After we discard the fixation trials (they contain no information relevant to our task), we are left with a total of 40 examples per subject, each example consisting of activity generated by both stimuli. While there is data available for multiple subjects,

there are difficulties in merging this data for the purpose of parameter estimation. This happens because different subjects have different brain shapes and because different subjects exhibit different intensity in activity when presented with the same cognitive task. Therefore we limited to analyzing data separately for each subject. The results reported in this section are all based on the same subject (04847). For this particular subject, our dataset tracked the activity of 4698 voxels.

We consider that the activity in each voxel is described by a Hidden Process Model with two processes, corresponding to the two stimuli: a *Sentence* process and a *Picture* process. The starting time of these processes are known in advance, given the structure of the trials described above. In half of the trials, the *Sentence* process starts at time 1 and in the other half it starts at time 17. The same holds for the *Picture* process. We make the assumption that the activity in different voxels is independent given the hidden processes corresponding to these voxels.

In our experiments we compare three models. Since the true underlying distribution is not available in this case, we use the Average Log Score to assess performance. Because the data is scarce, we can not afford to keep a held-out testing set. Therefore we employ a leave-two-out cross-validation approach to estimate the performance of our models. First model *StHPM*, which we will consider as a baseline, consists of a standard Hidden Process Model learnt independently for each voxel. The second model *ShHPM* is a Hidden Process Model, shared for all the voxels in an ROI. In other words, all voxels in a specific ROI share the same shape hidden processes, but with different amplitudes. *ShHPM* is learned using the algorithm presented in the previous subsection.

With only 40 training examples, the task of estimating the parameters can prove more than challenging. Therefore we would definitely benefit from an expert’s domain knowledge saying which groups of neighboring voxels are described by a Shared Hidden Process Model. We have seen that *ShHPM* makes the assumption that each ROI is a Shared Hidden Process Model. However, this assumption might not always be true. In the absence of a domain expert, we propose an algorithm which allows us to both automatically discover clusters of voxels that form a Shared Hidden Process Model and estimate the corresponding parameters. This third model (*HieHPM*) uses a nested cross-validation hierarchical approach to both come up with a partition of the voxels in clusters that form a Shared Hidden Process Model and estimate its corresponding performance on examples not used in training:

Algorithm (Hierarchical Partitioning and Hidden Process Models learning)

STEP 1. Split the 40 examples in a set of 20 folds $F = \{F_1, \dots, F_{20}\}$, each fold containing one example where the sentence is presented first and an example where the picture is presented first.

STEP 2. For all $1 \leq k \leq 20$, keep fold F_k aside and learn a model from the remaining folds using Steps 3-5.

STEP 3. Start with a partition of all voxels in the brain by their ROIs and mark all subsets as *Not Final*.

STEP 4. While there are subsets in the partition that are *Not Final*, take any such subset and try to split it using equally spaced hyperplanes on all three directions (in our experiments we split each subset in 2 by 2 by 4 smaller subsets). If the cross-validation Average Log Score of the model learnt from these new subsets using the algorithm presented in subsection 3.1 (based on folds $F \setminus F_k$) is lower than the cross-validation Average Log Score of the initial subset for folds in $F \setminus F_k$, then mark the initial subset as *Final* and discard its subsets. Otherwise remove the initial subset from the partition and replace it with its subsets which then mark as *Not Final*.

STEP 5. Given the partition computed by STEPS 3 and 4, based on the 38 data points in $F \setminus F_k$, learn a Hidden Process Model that is shared for all voxels inside each subset of the partition. Use this model to compute the log score for the examples/trials in F_k .

STEP 6. In Steps 2-4 we came up with a partition for each fold F_k . To come up with one single model, compute a partition using STEPS 3 and 4 based on all 20 folds, then, based on this partition learn a model as in STEP 5 using all 40 examples. The Average Log Score of this last model can be estimated by averaging the numbers obtained in STEP 5.

3.3 Results and Discussion

We estimated the performance of our models using the Average Log Score based on a leave two out cross-validation approach, where each fold contains an example where the sentence is presented first and an example where the picture is presented first.

Our first set of experiments, summarized in Table 1, compared the three models based on their performance in the Visual Cortex (CALC). This is one of the ROIs actively involved in this cognitive task and contains 318 voxels. The training set size was varied from 6 examples to all 40 examples, in multiples of two. As in the previous sections, sharing parameters of Hidden Process Models proved very beneficial and the impact was observed best when the training set size was the smallest. With an increase in the number of examples, the performance of *ShHPM* starts to degrade because it makes the biased assumption that all voxels in CALC can be described by a single Shared Hidden Process Model. While this assumption paid off with small training set size because of the reduction in variance, it definitely hurt in terms of bias with larger sample size. Even though the bias was

Training Trials	No Sharing (StHPM)	All Shared (ShHPM)	Hierarchical (HieHPM)	Cells (HieHPM)
6	-30497	-24020	-24020	1
8	-26631	-23983	-23983	1
10	-25548	-24018	-24018	1
12	-25085	-24079	-24084	1
14	-24817	-24172	-24081	21
16	-24658	-24287	-24048	36
18	-24554	-24329	-24061	37
20	-24474	-24359	-24073	37
22	-24393	-24365	-24062	38
24	-24326	-24351	-24047	40
26	-24268	-24337	-24032	44
28	-24212	-24307	-24012	50
30	-24164	-24274	-23984	60
32	-24121	-24246	-23958	58
34	-24097	-24237	-23952	61
36	-24063	-24207	-23931	59
38	-24035	-24188	-23921	59
40	-24024	-24182	-23918	59

Table 1: The effect of training set size on the Average Log Score of the three models in the Visual Cortex (CALC) region.

obvious in CALC, we will see in other experiments that in certain ROIs, this assumption holds and in those cases the gains in performance may be pretty big.

As expected, the hierarchical model *HieHPM* performed better than both *StHPM* and *ShHPM* because it takes advantage of Shared Hidden Process Models while not making the restrictive assumption of sharing across whole ROIs. The highest difference between *HieHPM* and *StHPM* is observed at 6 examples, in which case *StHPM* basically fails to learn a "decent" model while the highest difference between *HieHPM* and *ShHPM* happened with the maximum number of examples, when *ShHPM* started to be hurt by its bias. As the amount of training data increases, both *StHPM* and *HieHPM* tend to perform better and better and one can see that the difference in performance given by the addition of two new examples tends to shrink as both models approach convergence. While with infinite amount of data, one would expect *StHPM* and *HieHPM* to converge to the true model, at 40 examples, *HieHPM* still outperforms the baseline model *StHPM* by a difference

of 106 in terms of Average Log Score, which translates to an improvement of e^{106} in terms of data likelihood.

Probably the measure that shows best how much better is *HieHPM* than the baseline *StHPM* is given by how many more examples *StHPM* needs to achieve the same performance as *HieHPM*. It turns out that on the average, *StHPM* needs roughly 2.9 times more examples in order to perform same as well as *HieHPM* in the Visual Cortex (CALC).

The last column of Table 1 displays the number of clusters of voxels in which *HieHPM* partitioned CALC. As one may notice, at small sample size, *HieHPM* draws its performance from gains in variance by using only one cluster of voxels. However, as the amount of data increases, *HieHPM* improves by finding more and more refined partitions. This number tends to stabilize around 60 clusters once the number of examples reaches 30, which means an average of more than 5 voxels per cluster given that CALC is made of 318 voxels. For a training set of 40 examples, the largest cluster has 41 voxels while a lot of clusters are made of only one voxels.

The second set of experiments (see Table 2) describes the performance of the three models on all 24 ROIs of the brain as well on all the brain. While we have seen that *ShHPM* was biased in CALC, we may see here that there are several ROIs where it makes sense to characterize all voxels by a Shared Hidden Process Model. In fact, in most of these regions, *HieHPM* finds only one cluster of voxels. Actually, *ShHPM* outperforms the baseline model *StHPM* in 18 out of 24 ROIs while *HieHPM* outperforms *StHPM* in 23 ROIs. One may ask how can possibly *StHPM* outperform *HieHPM* on a ROI, since *HieHPM* may also represent the case when there is no sharing? The explanation is that the hierarchical approach can get stuck in a local maximum of the data log-likelihood over the search space if it cannot improve by splitting at a specific step since it does not look beyond that split for a finer grained partition. Fortunately, this problem is extremely rare, as we have seen in our experiments.

Over the whole brain, *HieHPM* outperforms *StHPM* by a factor of e^{1792} in terms of data likelihood while *ShHPM* outperforms *StHPM* only by a factor of e^{464} . However, the main drawback of the *ShHPM* is that it can be biased and therefore our experiments recommend *HieHPM* as the clear winner. Next we are going to give the reader a feel of what the model *HieHPM* looks like.

As mentioned above, *HieHPM* automatically learns clusters of voxels that can be represented using a Shared Hidden Process Model. Figure 2 shows the portions of these learned clusters in slice five of the eight vertical slices of the image of the brain taken by the fMRI scanner. Neighboring voxels that were assigned by *HieHPM* to the same cluster are pictured with the same color. Note that there are several very large clusters in this picture. This may be because of the fact that it makes sense to represent whole ROIs using a Shared Hidden Process Model if the studied cognitive task does not involve those areas of the brain. However, large clusters are also found in areas like CALC, which we know is directly involved in any visual activity.

ROI	Voxels	No Sharing (StHPM)	All Shared (ShHPM)	Hierarchical (HieHPM)	Cells Hierarchical
CALC	318	-24024	-24182	-23918	59
LDLPFC	440	-32918	-32876	-32694	11
LFEF	109	-8346	-8299	-8281	6
LIPL	134	-9889	-9820	-9820	1
LIPS	236	-17305	-17187	-17180	8
LIT	287	-21545	-21387	-21387	1
LOPER	169	-12959	-12909	-12909	1
LPPREC	153	-11246	-11145	-11145	1
LSGA	6	-441	-441	-441	1
LSPL	308	-22637	-22735	-22516	4
LT	305	-22365	-22547	-22408	18
LTRIA	113	-8436	-8385	-8385	1
RDLPFC	349	-26390	-26401	-26272	40
RFEF	68	-5258	-5223	-5223	1
RIPL	92	-7311	-7315	-7296	11
RIPS	166	-12559	-12543	-12522	20
RIT	278	-21707	-21720	-21619	42
ROPER	181	-13661	-13584	-13584	1
RPPREC	144	-10623	-10558	-10560	1
RSGA	34	-2658	-2654	-2654	1
RSPL	252	-18572	-18511	-18434	35
RT	284	-21322	-21349	-21226	24
RTRIA	57	-4230	-4208	-4208	1
SMA	215	-15830	-15788	-15757	10
All Brain	4698	-352234	-351770	-350441	299

Table 2: Per ROI performance of the three models when learned using all 40 examples.

In Figure 3 we can see the learned *Sentence* hidden process for the voxels in the Visual Cortex (CALC). Again, the graphs corresponding to voxels that belong to the same cluster have been painted in the same color, which is also the same with the color used in Figure 2. To make these graphs readable, we only plotted the base process, disregarding the scaling (amplitude) constants corresponding to each voxel within a given cluster.

To summarize, in this section we learned three different generative models for the fMRI signal

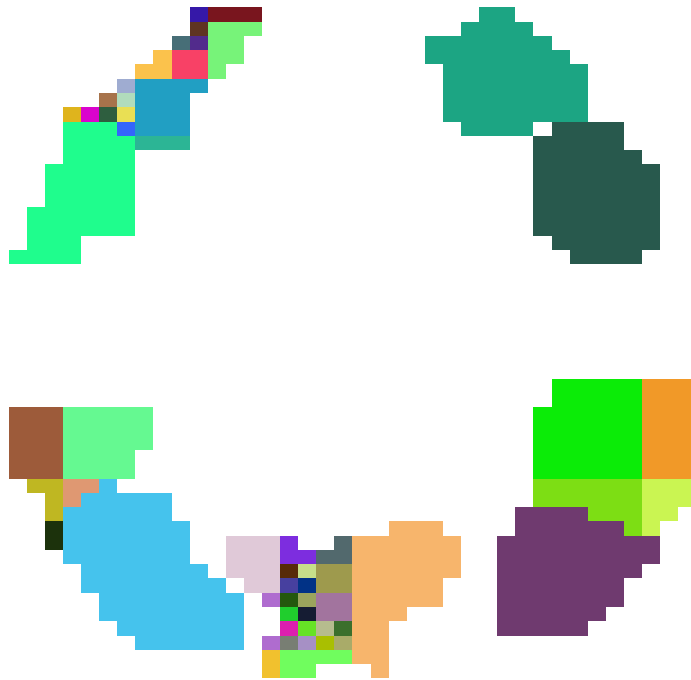


Figure 2: Parameter Sharing found using model *HieHPM*. Slice five of the brain is showed here. Shared neighboring voxels have the same color.

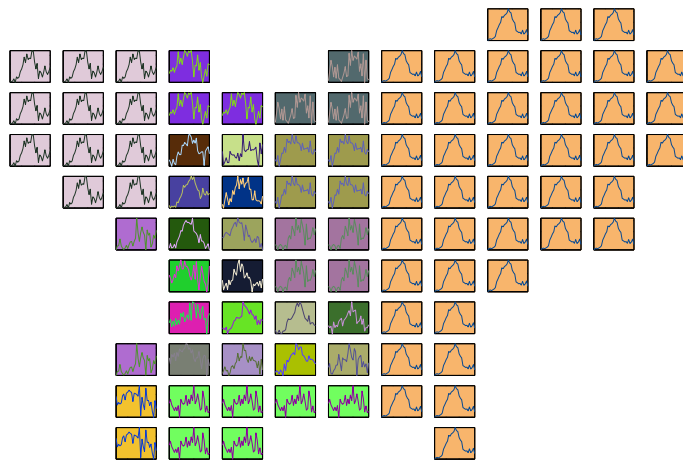


Figure 3: Per voxel base *Sentence* processes in the Visual Cortex(CALC).

during a cognitive task, all based on Hidden Process Models. We proved that Parameter Sharing for Hidden Process Models (as defined in Section) can greatly benefit learning. Our hierarchical

HieHPM model outperformed the other two models because it is both unbiased and able to reduce variance in the parameter estimators by automatically finding clusters of voxels that can be described by a Shared Hidden Process Model.

4 Summary of Thesis Results

Building accurate models from a small amount of available training data can sometimes prove to be a great challenge. Expert domain knowledge can be often used to alleviate this burden. In this thesis we presented the basis of a sound mathematical framework for incorporating Parameter Domain Knowledge in learning procedures for Bayesian Networks. We proved both theoretically and experimentally that the standard methods of performing parameter estimation in Bayesian Networks can be naturally extended to take advantage of Parameter Domain Knowledge that can be provided by a domain expert.

The most important contribution of this thesis was the development of a unified framework for incorporating general Parameter Domain Knowledge constraints in estimators for the parameters of a Bayesian Network by phrasing the goal as a constraint optimization problem. We showed how to compute Maximum Likelihood estimators using an iterative procedure based on Newton-Raphson method. This procedure can be computationally expensive, but fortunately, in practice, the optimization problem can be broken down into a set of many smaller, independent, optimization subproblems. Then, we discussed how to define Constrained Parameter Priors and perform learning from a Bayesian point of view. We also demonstrated how our methods can be extended in the case when the data is partially observable.

The iterative procedure mentioned above can be quite expensive. Therefore, it is preferable to derive closed form solutions for our estimators. This is not possible for general domain knowledge constraints but fortunately it is possible for several types of constraints, including parameter sharing of different kinds, as well as relationships among groups of parameters. Table 3 summarizes the results that we derived for these types of Parameter Domain Knowledge. Examples for each specific type of domain knowledge was described where that type was introduced. We approached learning of both discrete and continuous variables, in the presence of both equality and inequality constraints. While for most of these types of Domain Knowledge we can derive closed form Maximum Likelihood estimators, we come up with a very efficient iterative algorithm to perform the same task for Shared Hidden Process Models. In many of these cases, for discrete variables, we are also able to compute closed form normalization constants for the corresponding Constrained Parameter Priors, which allows us to perform closed form MAP and Bayesian estimation when the data is complete. We want to point out here that our General Parameter Sharing Framework can encompass models including HMMs, Dynamic Bayesian Networks, Module Networks and Context

Specific Independence as particular cases, but allows for much finer grained sharing, at parameter level, across different variables and across distributions of different lengths. It is also important to note that we can mix different types of Parameter Domain Knowledge constraints when learning the parameters of a Bayesian Network as long as the scopes of these constraints do not overlap.

Experimental results on fMRI data proved that taking advantage of domain knowledge can be very beneficial for learning. Since the domain knowledge was not always readily available, we developed methods to automatically uncover this knowledge. Using these methods we discovered clusters of voxels that can be learned together using Shared Hidden Process Models. Our results showed that the effect of the learned Parameter Domain Knowledge can be equivalent to almost tripling the size of the training set on this task. This was a pessimistic estimate of the benefits since we had to extract the domain knowledge from the training data itself via the cross-validation approach described in section 3. Experiments on synthetic data were also performed and they exhibited the same beneficial effect of incorporating Parameter Domain Knowledge.

A very important result that we managed to prove was that the estimators taking advantage of a simple form of Parameter Sharing achieved total variance lower than the one of estimators that ignored such domain knowledge. We conjecture that similar results hold for other types of domain knowledge, but their proof is left as future work.

In all the approaches above, we assumed the domain knowledge is correct. However, even when the domain expert makes mistakes, we proved that, with infinite amount of data, our Maximum Likelihood estimators would converge to the "best distribution" (the closest in terms of KL distance from the true distribution) that obeys the expert's assumptions and factorizes according to the given structure.

This research provided a new perspective of looking at learning procedures in the presence of domain knowledge about relationships among parameters. We feel that there is a lot of room for additional improvement in this exciting area of Parameter Domain Knowledge, an area which was barely explored so far.

Parameter Domain Knowledge Type	Results
Known Parameters, Discrete	Closed Form MLEs, MAP, Normalization Constant
Parameter Sharing, One Distribution, Discrete	Closed Form MLEs, MAP, Normalization Constant
Proportionality Constants, One Distribution, Discrete	Closed Form MLEs, MAP, Normalization Constant
Sum Sharing, One Distribution, Discrete	Closed Form MLEs
Ratio Sharing, One Distribution, Discrete	Closed Form MLEs
General Parameter Sharing, Multiple Distributions, Discrete	Closed Form MLEs, MAP, Normalization Constant
Hierarchical Parameter Sharing, Multiple Distributions, Discrete	Closed Form MLEs, MAP, Normalization Constant
Sum Sharing, Multiple Distributions, Discrete	Closed Form MLEs
Ratio Sharing, Multiple Distributions, Discrete	Closed Form MLEs
Inequalities between Sums of Parameters, One Distribution, Discrete	Closed Form MLEs
Upper Bounds on Sums of Parameters, One Distributions, Discrete	Closed Form MLEs
Parameter Sharing, One Distribution, Continuous	Closed Form MLEs
Proportionality Constants, One Distribution, Continuous	Closed Form MLEs
Parameter Sharing for Hidden Process Models	Efficient Iterative Method to Compute MLEs
Twice Differentiable with Continuous Second Derivatives	Iterative Methods: Frequentist, Bayesian, Complete and Incomplete Data

Table 3: Domain Knowledge Types studied in this thesis: description and results.